Synergetics and Theory of Chaos

УДК 167.7, 517

Gritsak von Groener V.V., Gritsak-Groener J., Petrou M.

BIOCOMPUTERS 3

University of Georgia, Georgia, USA; Laboratory of HRIT Corporation, Switzerland, USA, UK
Imperial College, University of London; QuVIET
e-mail: v_hrit1000000@yahoo. com

Biological computing theory has its roots in mathematical biology and mathematical computer sciences. Introduced by V. V. Gritsak-Groener [1, 9] in giving an example of superpower biological computational creation, which has beginning of new field of computer sciences. The aim of this article is to describe complex logical interaction between the DNA, RNA-molecule that is transmitted through the medium of the linear cell automata. We show (Theorem 1 and Theorem 2) that the interaction's structure of DNA-RNA is logician's equivalent to the structure Linear Cell Automata (g-LCA) and we show that g-LCA is Full Logical (Theorem 3 and Theorem 5). This article contains the main results: Theorems 6–7 determines the needed and the sufficient conditions of equivalence g-LCA to Universal Recursive Construct (URC). This article is continuous for articles [10, 11].

Keywords: computer, genome, cellular automata, matroid, category, DNA, RNA.

1. Introduction

Biological computing theory has its roots in mathematical biology and mathematical computer sciences. Introduced by first author (see [1], [9]) in giving an example of superpower biological computational creation, which has beginning of new field of computer sciences. The aim of this article is to describe complex logical interaction between the DNK, RNK-molecule that is transmitted trough the medium of the cell automata.

We shall recall briefly the notion of biological computing.

A Linear Cell Automata (g-LCA), when it goes beyond the very elementary level of the General Cell Automata, makes considerable use of the results of DNA-Automata, as we remarked in the p.3. Let f be given by

$$f\colon \{\text{Ala, Cys, . . ., Trp, Tyr}\} \longrightarrow \text{Codons}. \qquad (1)$$

We have linear cell automata C = (D, f, end, end2), where D is DNK-molecule, f is stand function of C, end is beginning information and end2 is finishing of information.

Simple directed graph, or simple digraph $\mathfrak{I}$ is an ordered pair:

$$\mathfrak{I} = (V(\mathfrak{I}), E(\mathfrak{I})),\ E(\mathfrak{I}) \subseteq V(\mathfrak{I}) \coprod V(\mathfrak{I}), \qquad (2)$$

where $V(\mathfrak{I})$ is a nonempty set called the set of vertices of $\mathfrak{I}$; $E(\mathfrak{I})$ is a set disjoint union from $V(\mathfrak{I})$, called the set of *arrows* of $\mathfrak{I}$. Before, if $e = \langle v_1, v_2 \rangle \in E(\mathfrak{I}) \Rightarrow e^{\#} = \langle v_2, v_1 \rangle \notin E(\mathfrak{I})$, $v_1, v_2 \in V(\mathfrak{I})$. If $e = \langle v_1, v_2 \rangle$ arrow of $\mathfrak{I}$, $v_1 \overset{def}{=} \partial^+ e$ is called the tail (or initial) of e, and $v_2 \overset{def}{=} \partial^- e$ are called the spike (or terminal) of e. A digraph Δ is an ordered pair

$$\Delta = (V(\Delta), E(\Delta)), \qquad (3)$$

where $E(\Delta) = E_1(\mathfrak{I}_1) \cup \ldots \cup E_n(\mathfrak{I}_n)$ and $E_i(\mathfrak{I}_i)$, $i = [1, n]$ is arrows of simple digraphs $\mathfrak{I}_i = (V(\Delta), E_i(\mathfrak{I}_i))$. Suppose $\Delta = (V(\Delta), E(\Delta))$ is a digraph. If $e = \langle v_1, v_2 \rangle \in E(\Delta)$, v_2 is called an outneighbor of v_1, and v_1 inneighbor of v_2. e is said to be incident out of v_1 and incident into v_2. $fl(v_1)$ (or $\delta^+ v_1$) denotes the set of inneighbors v_1 of in Δ. Similarly, $st(v_1)$ (or $\delta^- v_1$) denotes the set of outneighbors v_1 of in Δ. An arrow having the same ends is called a loop of Δ. A *diwalk* joining the vertex v_1 to the vertex v_{n+1} in a digraph Δ is an alternating sequence $L^{\rightarrow}$:

$$v_1\, e_1\, v_2\, e_2\, v_3 \ldots e_n\, v_{n+1} \qquad (4)$$

with e_i incident out of v_i and incident into v_{i+1}. A directed walk $L^{\rightarrow}$ (4) is called a *diloop* if $v_1 = v_{n+1}$. A digraph Δ (3) is called a *diforest* if Δ not contains a diloop. Moreover, a digraph

$$\Lambda = (V(\Lambda), E(\Lambda)), \qquad (5)$$

is called a *straightedge* if $V(\Lambda) = [0,1, 2, . . ., n]$ and $E(\Lambda) \subseteq \{(i-1, i)\}$, $i\in[1, n]$. Also, a *linear digraph* is called a digraph $\Gamma = (V, E)$ such that there exists a surjective map

$$\mathfrak{C}: \Gamma \longrightarrow \Lambda,$$

where Λ is a straightedge and a restriction of the map on diwalk $L^{\rightarrow}$ is injective map, where $L^{\rightarrow}$ is a subgraph Γ. Finally, a *vertex noun* of $v\in V$ is called $\mathfrak{C}(v)$. The others detailed see [5].

Let $A = (a_1, . . ., a_i, . . .) \neq \varnothing$ be a set, whose elements will be called *codons*. We say that triple $\mathfrak{J} = (A, \zeta, U)$ has *cell universe*, if there is given an injective map

$$\zeta :\{\#, A\} \longrightarrow U \cup U^2 ,$$

where U is non-empty set, $A = (a_1, . . ., a_i, . . .) \neq \varnothing$, and m is maximal arity of codons for A. The set U is called a *vertices cell* (v-*cell*). A c*ell digraph* Γ is an ordered pair

$$\Gamma = (V, E),\ V = A \setminus \{\#\},\ E \subset V \coprod V \tag{6}$$

where V is a nonempty set called the set of *vertices* of Γ; E is a subset disjoint union from V, called the set of arrows of Γ if the following conditions hold:

(1) $\forall x\in V, (x, x)\notin E$;

(2) $\forall x,y\in V, (x,y) \in \mathbf{E} \Rightarrow (y,x)\notin E$.

Before, if $e = <v_1, v_2>\in E \Rightarrow e^* = <v_2, v_1>\notin E$, $v_1,v_2 \in V$. If $e = <v_1, v_2>$ arrow of Γ, $v_1 \overset{def}{=} \partial^+e$ is called the *initial* of e, and $v_2 \overset{def}{=} \partial^-e$ are called the *terminal* of e.

Theorem 1. *Cell digraph* Γ of DNK-RNK -Automata is a linear diforest.

The proof's detailed is given in [6]

Let $\Gamma = (V, E)$ is cell digraph (6), $V^h \subseteq V$, $\#\in V^y \subseteq V$, and $D(\Gamma)$ is the set of all directed walk $l^{\rightarrow}(v_1, v_{n+1}) = v_1 e_1 v_2 e_2 v_3. . . e_n v_{n+1}$, where $v_1\in V^h$, $v_{n+1}\in V^y$. There is given a map

$$\psi: E \longrightarrow (\Xi, \Xi\times\Xi).$$

The set Ξ is called an *arrows cell* (a-*cell*). V^h and V^y are called an *input* and an *output*. Sekstant

$$CL = (\Gamma , \Xi , \psi , V^h, V^y, d) \tag{7}$$

is called *2 -Band Cellular Automata* (2CA). Further, the word $S = a_1 a_2 . . . a_s \#$ is induces 2LCA (7), if $a_1\in V^h$, $a_s\in V^y$, $\zeta\{\#\} \longrightarrow U\cup U^2$, $\zeta\{a_i\} \longrightarrow U\cup U^2$, $\forall(a_i,a_{i+1}) \in E$, $\psi(a_i,a_{i+1}) \longrightarrow (\Xi, \Xi\times\Xi)$. Here, CL is called *2 -Band Linear Cellular Automata* (2LCA).

Without loss of generality it can be assumed that 2LCA make up by the two cell bands, where DNA is the cell band (DNA-band) and RNA is the arrows cell band (RNA-band). Futhermore, an information of DNA-band takes to RNA-band under the structure of cell digraph Γ.

Theorem 2. 2 -Band Linear Cellular Automata DNK-RNK-Automata is local isomorphic to the Linear Cellular Automata (LCA), where LCA has N^g rules of cell transform and St^g cell states, $\mu(N^g) < \infty$ and $\mu(St^g) < \infty$.

The proof is given in [7], [8].

Corollary 2.1. 2LCA DNK-RNK-Automata $\mathfrak{I}$ is isomorfic to the linear cellular automata g-LCA. Let N^g and N^0 rules of cell transform of $\mathfrak{I}$ and g-LCA, St^g and St^0 cell states $\mathfrak{I}$ and g-LCA, then g-LCA has $N^g \leq 2N^0 +1$, $St^g \leq 2St^0$.

2. Logical Realization of g-LCA $\mathfrak{I}$

The *formal symbol* (*fs*) of a logical theory are the following:

(a) $A =\{a_1, a_2\}$ codon – *letters (c-letters)*.

(b) The logical sign "$\vee$ ", which is called the *disjunction*.

(c) The logical sign "⌉ ", which is called the *negation*.

(d) The logical sign "⊚", which is called the *distiguition*

(e) The logical sign "⊡", which is called the *replication*.

(f) The specific sign "=", which is called the *equation*.
(g) The specific sign "$\subset$", which is called the *includition*.
(h) The specific sign "$\in$", which is called the *belongution*.
(i) The specific sign "(", which is called the *left bracket*.
(j) The specific sign ")", which is called the *right bracket*.

In g-LCA $\Im$ letters $A = (a_1, \dots, a_i, \dots) \neq \varnothing$ are the codons, where $\mu(A) < \infty$. M(A) be the free monoid generated by A. The elements of M(A) are called g-*words* and are identified with finite sequences

$$S = a_1\, a_2 \dots a_s\, \#, \tag{8}$$

where $a_i \in A$, $i = [1,s]$, # = end or end2. We recall that the *length* l(S) of g-words (8) is s. The composition in M(A) will be written multiplicatively, so that

$$S_1 \circ S_2 = a_1\, a_2 \dots a_s\, \#\, b_1\, b_2 \dots b_r\, \#,$$

is the sequence obtained by juxtaposition of $S_1 = a_1\, a_2 \dots a_s\, \#$ and $S_2 = b_1\, b_2 \dots b_r\, \#$.

The 0-words $\varepsilon = \varnothing$ is the idedentity element of the monoid M(A). Without loss of generality it can be designate that

$$S_1 \circ S_2 = a_1\, a_2 \dots a_s\, b_1\, b_2 \dots b_r, \tag{9}$$

And designate of g-word S that

$$S = a_1\, a_2 \dots a_s.$$

Theorem 3. g-LCA $\Im$ is linear cell automata if 30 cell states are:

a) 20 non-empty codons of genetic code;
b) 7 formal symbols;
c) end-codon, end2-codon, $\varnothing$-cell,

and 8 rules of cell transform of $\Im$.

Proof. The proofs of all statements in this theorem, including all lemmas can be found in [6], [8].

Lemma 3.1. Let g-LCA $\Im$ is contained in the band the word $S = S_1 \circ S_{\Diamond}^{n} \circ S_2$, where $S_1 = a_1 \dots a_s$, $S_2 = b_1 \dots b_r$, $S_{\Diamond}^{n} = \varnothing_1, \dots, \varnothing_n$, $\forall \varnothing_i$ is equvalence to $\varnothing$-cell. Then there is an $\Im$−algorithm of the processing $S_1 \circ S_{\Diamond}^{n} \circ S_2 \longrightarrow S_1 \circ S_2$.

The Algorithm Scheme. We leave to the words S_1, S_2 /{$\varnothing_n$} without change. Further we change a cell $\varnothing_n$ on b_1, the cell b_1 on b_2, . . ., the cell b_{r-1} on b_r, see Fig. 1.

a_1	,	,	.	a_s	$\varnothing_1$	,	,	,	$\varnothing_n$	b_1	,	,	,	b_r

$\Downarrow$

.	,	,	a_1	,	.	,	a_s	b_1	.	.,	,	b_r	,	.

Fig. 1

The detailed $\Im$− algorithms of the processing $S_1 \circ S_{\Diamond}^{n} \circ S_2 \longrightarrow S_1 \circ S_2$ can be found in [6].

Lemma 3.2. Let S be a g-word. We shall denote by $\vee S$ obtained by writing, from left to right, the sign "$\vee$", the g-word S. Let $S_1 = a_1 \dots a_s$ and $S_2 = b_1 \dots b_r$ be g-words. Then there is an $\Im$−algorithm of the processing $S_1, S_2 \longrightarrow \vee(S_1 \circ S_2)$, where we shall denote by $\vee(S_1 \circ S_2)$, which is called a *disjunctions* of S_1 and S_2, is a g-word, see Fig. 2.

.	a_1	.	.	.	a_s	.	.	.	b_1	.	.	.	b_r	.

$\Downarrow\vee$

.	$\vee$	(	a_1	.	.	.	a_s	b_1	.	.,	.	b_r	)	.

Fig. 2

The detailed $\Im-$ algorithms of the processing $S_1,S_2 \longrightarrow \vee(S_1 \circ S_2)$ can be found in [6].

Lemma 3.3. Let $S = a_1 \dots a_s$ be a g-word. We shall denote by $\rceil(S)$ obtained by writing, from left to right, the sign "$\rceil$", which is called a *negations* of S (It is read : not S). Then there is an $\Im-$ algorithm of the processing $S \longrightarrow \rceil(S)$, see Fig. 3.

.	.	.	.	.	.	a_1	.	.	.	a_s	.	.	.	.

$\Downarrow\rceil$

.	.	.	.	$\rceil$	(	a_1	.	.	.	a_s	)	.	.	.

Fig. 3

The detailed $\Im-$ algorithms of the processing $S \longrightarrow \rceil(S)$ can be found in [6].

Lemma 3.4. Let $S = a_1 \dots a \dots a \dots a_s$ be a g-word and "a" be a letter. We shall denote by $\circledcirc_a S$ the g-word constructed as follows: form the g-word $\circledcirc S$, link each occurrence of a in S to the $\circledcirc$ written on the left of S, and then replace a everywhere it occurs by the sign $\boxdot$, which is called a *distiguitions* of S. Then there is an $\Im-$algorithm of the processing $S \longrightarrow \circledcirc_a(S) =$ $= a_1 \dots \boxdot \dots \boxdot \dots a_s$, see Fig. 4.

.	.	a_1	.	.	.	a	.	.	.	a	.	.	.	a_s

$\Downarrow\circledcirc_a$

.	.	a_1	.	.	.	$\boxdot$	.	.	.	$\boxdot$	.	.	.	a_s

Fig. 4

The detailed $\Im-$ algorithms of the processing $S \longrightarrow \circledcirc(a)(S)$ can be found in [8].

Lemma 3.5. Let $S_1 = a_1 \dots a_s$ and $S_2 = b_1 \dots b_r$ are g-words. We shall denote by $=(S_1 \circ S_2)$ obtained by writing in left of $S_1 \circ S_2$ the sign "=", when S_1 coincindence S_2, which is called an *equations* of S_1 to S_2 (It is read: S_1 equation to S_2), for otherwise it denote $=\rceil(S_1 \circ S_2)$ obtained by writing in left the signs "$=\rceil$". Then there is an $\Im-$algorithm of the processing $S_1,S_2 \longrightarrow = (S_1 \circ S_2)$ or $=\rceil(S_1 \circ S_2)$, see Fig. 5.

=	$\rceil$	(	a_1	.	.	.	a_s	b_1	.	.,	.	b_r	)	.

$\Uparrow=\rceil$

.	a_1	.	.	.	a_s	.	.	.	b_1	.	.	.	b_r	.

$\Downarrow=$

.	=	(	a_1	.	.	.	a_s	b_1	.	.,	.	b_r	)	.

Fig. 5

The detailed $\Im-$ algorithms of the processing $S_1,S_2 \longrightarrow = (S_1 \circ S_2)$ or $=\rceil(S_1 \circ S_2)$ can be found in [7], [8].

Lemma 3.6. Let $S_1 = a_1 \dots a_s$ and $S_2 = b_1 \dots b_r$ are g-words. We shall denote by $\subset(S_1 \circ S_2)$ obtained by writing in left of $S_1 \circ S_2$ the sign "$\subset$", when S_1 is a segment of S_2, which is called a *includations* of S_1 to S_2. Then there is an $\Im-$algorithm of the processing $S_1,S_2 \longrightarrow \subset(S_1 \circ S_2)$, see Fig. 6.

.	a_1	.	.	.	a_s	.	.	.	b_1	.	.	.	b_r	.

$\Downarrow\subset$

.	$\subset$	(	a_1	.	.	.	a_s	b_1	.	.,	.	b_r	)	.

Fig.6

The detailed $\Im-$ algorithms of the processing $S_1,S_2 \longrightarrow \subset(S_1 \circ S_2)$ can be found in [7], [8].

Lemma 3.7. Let $S = a_1 \ldots a_s$ is g-word and a is a c-letter. We shall denote by $\in(a\circ S)$ obtained by writing in left of $a\circ S$ the sign "$\in$", when a is a segment of S, which is called a *belongutions* of a to S. Then there is an $\Im-$algorithm of the processing $a,S \longrightarrow \in(a\circ S)$, see Fig. 7.

.	.	.	.	.	a	.	.	.	a_1	.	.	.	a_s	.

$\Downarrow\in$

.	.	.	.	.	$\in$	(	a	a_1	.	.,	.	a_s	)	.

Fig.7

The detailed $\Im-$ algorithms of the processing $a, S \longrightarrow \in(a\circ S)$ can be found in [7], [8].

Corollary 3.1. We have

$$|N^g| \le 27, |St^g| \le 10. \qquad (10)$$

Theorem 4. A linear cellular automata is universal computation.

Proof can be found in [7, 8].

3. g-LCA Universal Design

By [2] (detailed in [7], [8]), it follows that universal cellular automata is isomorphic g-LCA universal design. For all genetic information in system

$$DNR \longrightarrow RNR \longrightarrow Protein \qquad (11)$$

we have g-LCA universal design.

The specific signs "=", "$\subset$", and "$\in$", of a theory G are called an *iteration*, and the others are called a *linguisticotion*. With every the iteration is associated a natural number called its weight. A g-word is said to be of the *first genus* if it begins with a linguistically sign, or with a "$\circledcirc$", or if it consists of a c-letters; otherwise it is of the *second genus*.

Theorem 5. Let in the theory G of LCA is a sequences $S_1 \ldots S_n$ of g-words which has the following property: for each g-word S of the sequence, one of the following conditions is satisfied:

(1) S is a c-letter.

(2) There are two g-words S_1 and S_2 of the second genus, preceding S, such that S is $S_1 \vee S_2$.

(3) There is g-word S_1 of the second genus, preceding S, and a c-letter a such that S is $\circledcirc(a)S_1$.

(4) There is in the sequences a g-word S_1 of the second genus, preceding S, such that S is $\rceil S_1$.

(5) There is an iteration i of weight 2 in G, and two g-words S_1 and S_2 of the first genus, preceding S, such that is $i(S_1S_2)$.

Then the subsequences $S_1 \ldots S_i$, $i \le n$, is a *construct* of S_i.

Proof is in [2].

The g-words of the first genus, which appear in the construct of G, are called g-*terms* in LCA. The g-words of the second genus, which appear in the construct of G, are called g-*relations* in LCA.

4. Classified Theorem of 2LCA

We are given a map w: A ⟶ ***N***, where the set ***N*** is positive integers. $w(a_i)$ is called *arity* of the c-letter a_i. For each non-null the g-word S = a_1 a_2 . . . a_s, we put w(S) = $\sum_{i=1}^{s} w(a_i)$, and w(ε) = 0, w(#) = 2. w(S) is called the *mass* of the g-word S. We denote S⊡ the g-word obtained by deleting the states ⊡ in S with the left shift on the remote places. If S_1 = S '∘ S_2 ∘ S'', the g-word S_2 is said to be a segment of S_1.

A *worm* is a sequence S_i, i =[1,n], of g-words with the following property: for ∀ g-word S of sequence, one of the following two conditions is satisfied:

(1) S is a sign of mass 0.

(2) ∃ m (m < n) g-words S^1, . . ., S^m in the sequence such that it be founds in the worm S_i before S, and a sign φ mass m such that S = φ S^1 . . . S^m.

A *snake* is a g-word S of the following two conditions is satisfied:

(3) l(S) = w(S) + 1, where l(S) is length of the g-word S.

(4) For ∀ proper a segments S_1 of S, w(S_1) ≥ l(S_1).

Theorem 6. If a g-word S is an iteration and a linguisticotion in the theory G, then S is a snake.

Proof. The proofs of all statements in this theorem, including all lemmas can be found in [2], [8].

Lemma 6.1. If S_1, . . ., S_m are m a worm and if φ is a sign of mass m, then the g-word S = φS_1 . . . S_m is a worm.

Lemma 6.2. A g-word is a worm iff it is a snake.

Lemma 6.3. ∀ a worm may be представлено g-LCA in exactly one way in the form φS_1 . . . S_m, where S_1, . . ., S_m are worms and φ has mass m.

Theorem 7. Let a g-word S be a snake.

For S to be a g-term iff that one of the following conditions be satisfied:

(α) S consists of a single c-letter.

(β) S begins with "⊚", S⊡ is identical with the iteration φS_1 . . . S_m and its are g-relations.

(χ) S begins with a linguistication sign "θ", S⊡ is identical with the iteration θS_1 . . . S_m and its are g-terms.

For S to be a g-relation iff that one of the following conditions be satisfied:

(δ) S begins with a "∨" or a "⌉", S⊡ is identical with the iteration ∨S_1 . . . S_m (or ⌉S_1 . . . S_m) and its are g-relations.

(ε) S begins with the iteration sign "σ", S⊡ is identical with the iteration σS_1 . . . S_m and its are g-terms.

∀ S_i are a g-words.

Proof. The Lemma 7.1 – 7.4 show that the conditions of theorem 7 are sufficient.

Lemma 7.1. If S is a g-relation in the theory G of g-LCA, then ⌉S is g-relation in G.

Proof in [2] and [8].

Lemma 7.2. If S_1 and S_2 are g-relations in the theory G of g-LCA, then ∨ S_1S_2 is a g-relation in G.

Proof in [2] and [8].

Lemma 7.3. If S is a g-relation in the theory G of g-LCA, and if a is a c-letter, then $⊚_a$(S) is a g-term in G.

Proof in [2] and [8].

Lemma 7.4. If $S_1, \ldots, S_m$ are g-terms in the theory G of g-LCA, and if "σ" is an iteration of mass m in G, then $\sigma S_1 \ldots S_m$ is the g-relation in G.

Proof in [2] and [8].

Lemma 7.5. The conditions (α)-(ε) in theorem 7 are necessary conditions.

Then proof is trivial.

5. Hypothesis WILLIAM

Hypothesis WILLIAM. Is it true that the mechanism of transfer genetic information in system (10) is the classical recursive process?

Proposition. Hypothesis WILLIAM is said to be true if be realized the conditions theorems 6-7.

This will be the object of another paper.

References:

1. *Gritsak V. V.* (*Gritsak–Groener V. V.*) // Proc. Nat. Acad. Sci. Ukraine SSR, ser. A, N6(1990)68.
2. *Gritsak V. V.* (*Gritsak–Groener V. V.*) // Bulletin Schevtshenko Kyjiv National University, ser. phys.-math. science, N1(1993)5.
3. *Gritsak–Groener V. V.*, *Gritsak–Groener Julia, Arabnia Hamid R.* Superpower Computational Creation for Biological Computing // Proceedings of the 2005 International Conference on Scientific Computing, Las Vegas, Nevada, USA June 20-23. — 2005. — P. 184–190.
4. *Gritsak Valery V. (Gritsak–Groener V. V.), Okolita Julia P.* A Categorical Model of Neurosystem // Proceedings of the 1996 International Conference on Parallel and Distributed Processing Techniques and Applications, Sunnnyvale Hilton, California, USA, August 9-11. — 1996. — P. 1610–1613.
5. *Gritsak–Groener V. V., Gritsak–Groener Julia.* ARTS COMBINATORIA. Second edit., — Charkiv, NTU "HURE", 2008.
6. *Groener Wilhelm* (*Gritsak–Groener V. V.*) Fundamental of Mathematical Cybernetics. — Zhytomyr, ZhDTU, 2004.
7. *Gritsak–Groener V. V.*, *Hamid R. Arabnia*, Modern Mathematical Biology, MIT Press, N.Y., 2006.
8. *Gritsak–Groener V. V.* Fundamental of Mathematical Cybernetics, vol 2. — Zhytomyr, ZhDTU, 2006.
9. *Gritsak V. V.* Computational Model of Transfer RNA Molecules. Preprint 87IMBG-14. — Kyjiv, Institute of Molecular Biology and Genetics of National Academy Science of Ukraine, 1987.
10. *Gritsak–Groener V. V.* Biocomputers 1 // Physics of consciousness and life, cosmology and astrophysics. — 2010. — № 2.
11. *Gritsak–Groener V. V.* Haotics and Biocomputing Application (Biocomputers 2) // Physics of consciousness and life, cosmology and astrophysics. —2010. — № 3.

Статья поступила в редакцию 12.01.2011 г.

Гритсак-Грёнер В.В., Гритсак-Грёнер Ю., Петроу М.

Биокомпьютеры 3

Биологическая теория вычислений имеет свои корни в математической биологии и математической кибернетике. Найденный и продемонстрированный в [1, 9] V. V. Gritsak-Groener пример супермощных биологических вычислительных возможностей открывает новую область компьютерной науки. Цель этой статьи — описание действия ДНК и РНК-молекул посредством действия линейных клеточных автоматов. А так как в [4, 7] была получена полная классификация линейных клеточных автоматов, мы получаем полную классификацию логического взаимодействия ДНК, РНК-молекул. Показано (Теорема 1 и Теорема 2), что структура взаимодействия ДНК-РНК логически эквивалентна структуре специальных Линейных Клеточных Автоматов (g-LCA) и что g-LCA есть Логически Полные (Теорема 3 and Теорема Т 5). Основной результат статьи: Теоремы 6–7 о том, что g-LCA удовлетворяет необходимым и достаточным условиям для Универсальной Рекурсивной Конструкции (URC). Статья — продолжение [10, 11].
Ключевые слова: компьютер, геном, клеточный автомат, матроид, категория, ДНК, РНК.

Реферат

”Кошмар” современной компьютерной техники состоит в том, что, несмотря на увеличение ее быстродействия, нельзя гарантировано в приемлемое время решать NP-полные и более сложные задачи. (К NP-полным задачам принадлежат даже такие простейшие задачи, как задача коммивояжера, линейная одномерная задача оптимизации – одномерный «рюкзак» — и подобные). Более того, как было впервые показано в [1] и [3] задачи считывания и реализации генетической информации в ДНК, РНК взаимодействии есть, также NP-полной вычислительной задачей. Значит, любой живой организм обладает вычислительным процессором, который несравнимо превосходит по скорости вычислений любой сверхмощный современный и будущий компьютер (электронную реализацию классической рекурсивной теории).

Еще в 1929 году Джон фон Нейман показал, что плоский клеточный автомат может быть не менее мощным по вычислительной скорости, чем любой компьютер, реализующий любой вариант рекурсивной теории, например, Машину Поста.

В [1] и более подробно в [7], была дана полная классификация конечных линейных клеточных автоматов. В настоящей работе, показано, что ДНК, РНК-взаимодействие эквивалентно, некоторому специальному линейному клеточному автомату g-LCA (не более 27 состояний клеток и не более 10 вычислительных стрингов). Тем самым, предлагается проект биокомпьютера, который позволяет решать NP-полные задачи. Мы надеемся, что после нашей статьи приблизится конец «Кошмара» современной компьютерной техники.